Amendments to the Claims:

The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A computer implemented method comprising:

beginning initialization of a first thread from a second ~~context~~platform, wherein the first thread is executable on a first ~~context~~platform and the second ~~context~~platform, wherein the first thread is 32-bit platform-independent code;

suspending the initialization of the first thread at a position within the second ~~context~~platform, wherein the beginning initialization of the first thread is suspended in response to a detection of an operating system request to create the first thread;

creating a second thread based on the position in the second ~~context~~platform; ~~and~~

completing the initialization of the first thread continuing from the position in the second ~~context~~platform; and

translating the 32-bit platform-independent code for execution on the second platform, wherein the second platform is a 64-bit platform.

2. (Original) The method of claim 1, wherein the beginning initialization of the first thread includes allocating per-thread context resources.

3. (Canceled)

4. (Currently Amended) The method of claim 1, wherein the second ~~context~~platform is the 32-bit platform-independent code to be executed on a 64-bit host platform.

5. (Currently Amended) The method of claim 1, wherein the second ~~context~~platform is a host platform that supports multiple instruction set architectures (ISA).

6. (Currently Amended) The method of claim 1, wherein completing the initialization of the first thread includes executing an application programming interface that makes an operating system request to create the first thread in the second ~~context~~platform.

7. (Cancelled)

8. (Cancelled)

9. (Currently Amended) A computer implemented method comprising:

beginning initialization of a foreign thread from a host platform, wherein the foreign thread is to be executed on a foreign platform and the foreign thread is 32-bit platform independent code;

suspending the initialization of the foreign thread at a position within the host platform, wherein the beginning initialization of the foreign thread is suspended in response to a detection of an operating system request to create the foreign thread;

recording the position of the suspension;

creating a host thread from a host platform based on the recorded position; ~~and~~

completing the initialization of the foreign thread continuing from the recorded position in the host platform; and

translating the 32-bit platform-independent code for execution on the host platform, wherein the host platform is a 64-bit platform.

10. (Original) The method of claim 9, wherein the beginning initialization of the foreign thread includes allocating per-thread context resources.

11. (Canceled)

12. (Original) The method of claim 9, wherein the host platform supports platform-independent code.

13. (Original) The method of claim 9, wherein the host platform supports multiple instruction set architectures (ISA).

14. (Original) The method of claim 9, wherein the foreign platform is a IS-32 platform and the host platform is a IS-64 platform.

15. (Currently Amended) A computer system for managing thread resources comprising:

a memory;

a processor coupled with the memory;

a first multithreaded programming environment, executed by [[a]] the processor;

a second multithreaded programming environment, executed by the processor;

a multithreaded program, executed by the processor, including a first thread and a second thread, wherein the first thread is 32-bit platform independent code;

a host platform, wherein the host platform is a 64-bit platform; and

a dynamic binary translator to translate the first thread for the first multithreaded programming environment to be executed in the second multithreaded programming environment, wherein the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected, and wherein the translation includes translation of the 32-bit platform independent code for execution on the 64-bit platform.

16. (Previously presented) The system of claim 15 further comprising a first component to provide a communication interface between the dynamic binary translator and the multithread programming environment.

17. (Previously presented) The system of claim 15 further comprising a first thread library and a second thread library.

18. (Canceled)

19. (Currently Amended) A computer system for managing thread resource comprising:

a random accessed memory;

a first processor configured to execute multithreaded programs stored in the random accessed memory;

a multithreaded program to be executed on a second processor; and

a program to transparently initialize, ~~and~~ create, and translate a thread included in the multithreaded program in an environment supported by the first processor to be executed on the second processor, wherein the initialization of the thread is suspended in response to a detection of an operating system request to create the thread, and wherein the thread is 32-bit platform independent code and is translated for execution on a 64-bit platform.

20. (Original) The system of claim 19, wherein the program further allocates per-thread context resources.

21. (Original) The system of claim 20, wherein the program initializes and creates the thread transparently by associating the allocated per-thread context resource between the environment supported by the first processor.

22. (Canceled)

23. (Canceled)

24. (Canceled)

25. (Canceled)